# Learning Factored Markov Decision Processes with Unawareness

# Learning Factored Markov Decision Processes with Unawareness

**Craig Innes**
School of Informatics
University of Edinburgh
Edinburgh, United Kingdom EH8 9AB
craig.innes@ed.ac.uk

**Alex Lascarides**
School of Informatics
University of Edinburgh
Edinburgh, United Kingdom EH8 9AB
alex@inf.ed.ac.uk

## Abstract

Methods for learning and planning in sequential decision problems often assume the learner is aware of all possible states and actions in advance. This assumption is sometimes untenable. In this paper, we give a method to learn factored markov decision problems from both domain exploration and expert assistance, which guarantees convergence to near-optimal behaviour, even when the agent begins unaware of factors critical to success. Our experiments show our agent learns optimal behaviour on small and large problems, and that conserving information on discovering new possibilities results in faster convergence.

## 1 INTRODUCTION

Factored markov decision processes (FMDPs) are a fundamental tool for modelling complex sequential decision problems. When the transition and reward functions of an FMDP are known, there are tractable methods to learn its optimal policy via dynamic programming [Guestrin et al., 2003]. When these components are unknown, methods exist to jointly learn a structured model of the transition and reward functions [Degris et al., 2006, Araya-López et al., 2011]. Yet all such methods (except Rong [2016]) assume that the actions available to the agent and the state variables that describe the state space are completely known *in advance of learning*. In many scenarios, this assumption does not hold.

In human discussion for example, answers to a person's inquiry may not only provide information about which of the questioner's existing hypotheses are likely, but may also reveal new unconsidered hypotheses [Coenen et al., 2017]; in medicine, pharmacologists might discover that the side-effects of a previously tested drug are radically affected by a newly discovered gene; in robotic skill-learning, methods like Cakmak and Thomaz [2012] teach a robot how to *perform* new actions it was previously unaware of, but not how to *integrate* them into an existing decision model. In all these examples, it may be infeasible for an agent to gather all relevant factors of the problem *before* learning, but relatively easy for an expert to offer contextually relevant corrective advice *during* learning.

In previous work, Innes and Lascarides [2019] provide a framework for incremental learning under this kind of unawareness. Their framework uses evidence from both expert advice and domain trials to gradually expand an agent's awareness of the hypothesis space, rather than to refine a distribution over a fixed hypothesis space. This work dealt exclusively with simple *one-shot* decision tasks, where an agent's actions in the past do not have a lasting effect on future states. This paper directly extends such work to deal with *sequential* problems of potentially unbounded horizon.

One could just represent unawareness as an infinite number of hidden states [Doshi-Velez, 2009], or via a series of densely connected hidden layers [Mnih et al., 2015]. But these approaches have several drawbacks. First, neither currently addresses what to do when an unforeseen *action* is discovered. More importantly, since the hidden variables are not tied to grounded concepts with explicit meaning, it is difficult for an agent to justify its decisions to a user, or to articulate queries about its current understanding of the world so as to solicit help from an expert.

We instead propose a system where an agent makes explicit attempts to overcome its unawareness, extending Innes and Lascarides [2019] to handle sequential decision problems. This paper makes three contributions: First, an algorithm which incrementally learns *all* components of an FMDP. This includes the transition, reward, and value functions, but also the *set of actions and state variables themselves* (Section 3). Second, an expert-

agent communication protocol (Section 3.1) which interleaves contextual expert advice with learning, and guarantees our agent converges to near-optimal behaviour, despite beginning unaware of factors critical to success. Third, experiments on small and large sequential decision problems showing our agent successfully learns optimal behaviour in practice (Section 4).

## 2 THE LEARNING TASK

We focus on learning *episodic, finite state* FMDPs with discrete states and actions. We begin with the formalisms for learning optimal behaviour in FMDPs where the agent is fully aware of all possible states and actions. We then extend the task to one where the agent starts unaware of relevant variables and actions, and show how the agent overcomes this unawareness with expert aid.

### 2.1 EPISODIC MPDS

An MDP is a tuple $\langle \mathcal{S}, \mathcal{S}_s, \mathcal{S}_e, A, \mathcal{T}, \mathcal{R} \rangle$, where $\mathcal{S}$ and $A$ are the set of states and actions; $\mathcal{S}_s, \mathcal{S}_e \subseteq \mathcal{S}$ are possible start and end (terminal) states of an episode; $\mathcal{T} : S \times A \times S \to [0,1]$ is the *markovian transition function* $P(s'|s,a)$, and $\mathcal{R} : S \to \mathbb{R}$ is the immediate reward function.[1] A policy $\pi : \mathcal{S} \times A \to [0,1]$ gives the probability $\pi(s,a)$ that an agent will take action $a$ in state $s$. When referring to the local time $m$ in episode $n$, we denote the current state and reward by $s_{m,n}$ and $r_{m,n} = \mathcal{R}(s_{m,n})$. When referring to the *global time* $t$ across episodes, we denote them by $s_t$ and $r_t$.

The *discounted return* for episode $n$ is: $G^n = \sum_{i=0}^{T} \gamma^i * r_{i,n}$, where $0 \leq \gamma \leq 1$ is the *discount factor* governing how strongly the agent prefers immediate rewards. The agent's goal is to learn the *optimal policy* $\pi_+$, which maximizes the expected discounted return in all states. The *value function* $V_\pi(s)$ defines the expected return when following a given policy $\pi$, while the related action-value function $Q_\pi(s,a)$ gives the expected return of taking action $a$ in state $s$, and thereafter following $\pi$.

$$V_\pi(s) = \mathcal{R}(s) + \gamma \sum_{s' \in S} P(s'|s, \pi(s)) V_\pi(s') \quad (1)$$

$$Q_\pi(s,a) = \mathcal{R}(s) + \gamma \sum_{s' \in S} P(s'|s,a) V_\pi(s') \quad (2)$$

If $\mathcal{T}$ and $\mathcal{R}$ are known, we can compute $\pi_+$ via *value iteration* [Sutton and Barto, 1998]. Further, we can measure the expected loss in discounted return of following policy $\pi$ versus $\pi_+$ using (3), which we refer to as the *policy error*. If the agent's policy is unknown, we can approximate the policy error using (4), where $P(s_0)$ is the probability of starting an episode in $s_0$:

$$Err(\pi) = \sum_{s_0 \in \mathcal{S}_s} P(s_0)(V_{\pi_+}(s_0) - V_\pi(s_0)) \quad (3)$$

$$Err(t, t+k) = (\sum_{s_0 \in \mathcal{S}_s} P(s_0) V_{\pi_+}(s)) - \frac{\sum_{i=t}^{t+k} G^i}{k} \quad (4)$$

If all episodes eventually terminate, then (4) converges to (3). If our agent is *$\epsilon$-greedy* (i.e., in all states, has probability $\epsilon > 0$ of executing any action from $A$ at random), then termination in most MDPs is guaranteed [Sutton and Barto, 1998]:

**Definition 1** (Proper Policy)**.** *A policy $\pi$ is proper if, from all states $s \in \mathcal{S}$, acting according to $\pi$ guarantees one eventually reaches some terminal state $s' \in \mathcal{S}_e$.*

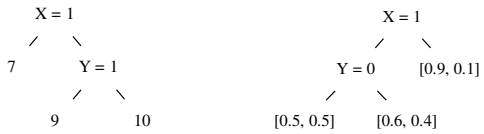
**Lemma 1.** *If an* MDP *has a proper policy $\pi$, then any policy which is $\epsilon$-greedy with respect to $A$ is also proper.*

### 2.2 LEARNING FMDPS WHEN FULLY AWARE

If $\mathcal{T}$ or $\mathcal{R}$ are unknown, the agent must learn them using the data $D_{0:t} = [d_0, \ldots, d_t]$ gathered from domain interactions. At time $t$, the *sequential trial* $d_t = \langle s_t, a_t, s_{t+1}, r_{t+1} \rangle$ gives the current state $s_t$, action $a_t$, resulting state $s_{t+1}$ and the reward $r_{t+1}$ given on entering $s_{t+1}$. FMDPs allow one to learn $\mathcal{T}$ for large MDPs by representing states as a *joint assignment* to a set of variables $\mathcal{X} = \{X_1, \ldots, X_n\}$ (written $\mathcal{S} = v(\mathcal{X})$). Similarly, the reward function is defined as a function $\mathcal{R} : v(scope(\mathcal{R})) \to \mathbb{R}$, where $scope(\mathcal{R}) \subseteq \mathcal{X}$ are variables which determine the reward received in each state. To exploit conditional independence, $\mathcal{T}$ is then represented by a *Dynamic Bayesian Network* (DBN) [Dean and Kanazawa, 1989] for each action. That is, $\mathcal{T} = \{dbn_{a_1}, \ldots dbn_{a_n}\}$, where $dbn_a = \langle Pa^a, \theta_a \rangle$. Here, $Pa^a$ is a directed acyclic graph with nodes $\{X_1, \ldots, X_n, X'_1, \ldots X'_n\}$ where, as is standard, node $X_i$ denotes the value of variable $X_i \in \mathcal{X}$ at the current time, while $X'_i$ denotes the same variable in the next time step. For each $X'_i$, $Pa^a_{X'_i}$ defines the parents of $X'_i$. These are the only variables on which the value of $X'_i$ depends. We also make the common assumption that values of variables which occur at exactly the same time cannot depend on one another. This means our DBNs contain no *synchronic arcs* [Degris and Sigaud, 2010], or formally that $\forall X'_i \, \forall a, Pa^a_{X'_i} \subseteq \{X_1, \ldots, X_n\}$.

This structure allows us to write transition probabilities

[1] In this paper, we assume the agent's preferences depend only on the current state, and are both *deterministic* and *stationary*. Other works allow $\mathcal{R}$ to depend on the action and/or resulting state (i.e., $\mathcal{R} : S \times A \times S \to \mathbb{R}$).




(a) Reward (b) Conditional Probability

Figure 1: Example decision trees

as a product of independent parameters: $P(s'|s,a) = \prod_{X \in \mathcal{X}} \theta^a_{s'[X], s[Pa^a_{X'}]}$. Here, $s[\vec{Y}]$ is the projection of $s$ onto the variables in $\vec{Y}$, and $\theta^a_{X'=i,Pa^a_{X'}=j}$ denotes the probability of variable $X$ taking on value $i$ given that the agent performs action $a$ when the variables $Pa^a_{X'}$ have assignment $j$ in the current time step.[2]

Exploiting independence among state variables doesn't guarantee a compact representation of $V_\pi$. We must also exploit the *context-specific* independencies between assignments by representing $\mathcal{T}$ and $\mathcal{R}$ as *decision trees*, rather than tables of values [Boutilier et al., 1996].

Figure 1 shows an example decision tree for $\mathcal{R}$ and $P(X'|X,Y)$. The *leaves* are either rewards, or a distribution over the values of $X'$. The non-leaves are *test nodes*, which perform a binary test of the form $(X = i?)$ to check whether variable $X \in \mathcal{X}$ takes on the value $i$ in the current state. Notice that when $X = 1$ is true, the distribution over $X'$ is conditionally independent of $Y$.

As in Innes and Lascarides [2019], we estimate the most likely DBN structure from $D_{0:t}$ using the *Bayesian Dirichlet Score* (5), with a prior (6) which penalizes complex DBNs with a cost $\rho < 0.5$ for each parent:

$$P(Pa^a_{X'}) \prod_{j \in v(Pa^a_{X'})} \frac{\beta(N^a_{1,j} + \alpha^a_{1,j}, \ldots, N^a_{m,j} + \alpha^a_{m,j})}{\beta(\alpha^a_{1,j}, \ldots, \alpha^a_{m,j})} \quad (5)$$

$$P(Pa^a_{X'}) = \rho^{|Pa^a_{X'}|}(1-\rho)^{|\mathcal{X}|-|Pa^a_{X'}|} \quad (6)$$

Here, $N^a_{X=i,Pa^a_{X'}=j}$ is the number of trials in $D_{0:t}$ where action $a$ was taken in a state where the joint assignment to $Pa^a_{X'}$ was $j$, resulting in a state where $X$ has the value $i$. The $\alpha$ terms are the hyper-parameters from the prior over the parameters, and act as "pseudo-counts" when data is sparse. Note, if the space of possible DBNs is too large, we can restrict the parent sets considered reasonable by using common pruning heuristics or, for example, restricting the maximum in-degree.

Given $Pa^a_{X'}$, we can then compute each variable's most likely conditional probability tree structure $\text{DT}^a_X$, restricting node tests to members of $Pa^a_{X'}$:

$$P(\text{DT}^a_X | Pa^a_{X'}, D_{0:t}) \propto P(\text{DT}^a_X) P(D_{0:t} | \text{DT}^a_X) \quad (7)$$

$$P(D_{0:t}|\text{DT}^a_X) = \prod_{\ell \in Leaves(\text{DT}^a_X)} \frac{\beta(N^a_{1|\ell} + \alpha^a_{1|\ell}, \ldots, N^a_{m|\ell} + \alpha^a_{m|\ell})}{\beta(\alpha^a_{1|\ell}, \ldots, \alpha^a_{m|\ell})} \quad (8)$$

Here, $N_{i|\ell}$ is the number of trials where $X = i$, and its parents have an assignment which matches against the branch label $\ell$. Rather than evaluating the probabilities of all possible DT structures at each step, we can incrementally update the most likely DT as new trials arrive using *incremental tree induction* (ITI), as described in Utgoff et al. [1997]. While we lack the space to describe ITI in detail here, the algorithm broadly works by maintaining a single most-likely tree structure, with counts for all potential test assignments cached at intermediate nodes. As new trials arrive, the counts at relevant nodes become "stale", as there might now exist an alternative test which could replace the current one, resulting in a higher value for equation (7). If such a superior test exists, the test at this node is replaced, and the tree structure is transposed to reflect this change. We can also use ITI to learn a tree structure for $\mathcal{R}$ based on the trials seen so far. The only difference is that we use an information-gain metric to decide on the best test nodes, rather than (7). For example, if our current reward tree looks like the one in figure 1a, and we receive a new trial where $s_{t+1} = \langle X = 1, Y = 1 \rangle$ and $r_{t+1} = 8$, then we would expand the left child of the $X = 1$ branch into a test node which tests for the value of $Y = 1$.

Finally, given $\text{DT}^a_X$, we compute the most likely parameters at each leaf via (9), where $N^a_{\cdot,j} = \sum_{i \in v(X)} N^a_{i,j}$:

$$\mathbb{E}(\theta^a_{X=i,Pa^a_{X'}=j} | D_{0:t}, \text{DT}^a_X) = \frac{N^a_{i,j} + \alpha^a_{i,j}}{N^a_{\cdot,j} + \alpha^a_{\cdot,j}} \quad (9)$$

Once our agent has a transition and reward tree, we can then use *structured value iteration* (SVI) [Boutilier et al., 2000]—a variant of value iteration which works with decision trees instead of tables—to compute a compact representation of $V_{\pi_+}$. Algorithm 1 shows an outline of an incremental version of SVI (iSVI) [Degris et al., 2006], which allows the agent to gradually update its beliefs about the optimal value function in response to incoming trials. The algorithm takes the current estimate of the reward and transition functions ($\mathcal{R}_t$ and $\mathcal{T}_t$), along with the previous estimate of the optimal value function ($V_{t-1}$), and combines them to produce a new estimate for

[2] If the context is clear, we condense this notation to $\theta^a_{i,j}$.

each state-action function ($Q_t^a$), and value function ($V_t$). For further details about the merge and regress functions used in SVI, consult Boutilier et al. [2000].

**Algorithm 1** Incremental SVI [Degris et al., 2006]

**function** INCSVI($\mathcal{R}_t$, $\mathcal{T}_t$, $V_{t-1}$)
  $\forall a \in A : Q_t^a \leftarrow \text{REGRESS}(V_{t-1}, dbn_a, \mathcal{R}_t)$
  $V_t \leftarrow \text{MERGE}(\{Q_t^a : \forall a \in A\})$ (using maximization as the combination function)
  **return** $\{V_t, \{\forall a \in A : Q_t^a\}\}$

This section took an *encapsulated* approach to learning $\mathcal{T}$ (In contrast to a *unified* one in e.g., Degris et al. [2006]). This means we separate the task of finding an optimal DBN structure from the task of learning each local DT structure. Such an approach significantly reduces the space of DTs that must be considered, but more importantly, provides us with posterior *distributions* $P(Pa_{X'}^a | D_{0:t})$ over parent structures. We will use these posterior distributions in section 3.2 to conserve information when discovering unforeseen factors.

# 3 OVERCOMING UNAWARENESS

So far, we've assumed our agent was aware of all relevant state variables in $\mathcal{X}$, all actions $A$, and all members of $scope(\mathcal{R})$. We now drop this assumption. From here onward we denote the true set of state variables, actions, and reward scope as $\mathcal{X}^+$, $A^+$ and $scope_+(\mathcal{R})$, and the learner's awareness of them at $t$ as $\mathcal{X}^t$, $A^t$, and $scope_t(\mathcal{R})$

Suppose $X^+ = \{X_1, X_2, X_3\}$, $X^0 = \{X_1\}$, $A^+ = \{a, a'\}$, $A^0 = \{a\}$. We assume the agent can't observe the value of variables it is unaware of. In the medical example from before, if $X_3$ corresponds to a particular gene, then we assume the agent cannot detect the presence or absence of that gene if it is unaware that it exists. Similarly, we assume the agent cannot perform an action it is unaware of.[3] As a consequence, at time $t = 0$, the agent does not directly observe the true trial $d_0$, but rather $d_0[\mathcal{X}^0] = \langle s_0[\mathcal{X}^0], a_0, s_1[\mathcal{X}^0], r_0 \rangle$. The key point here is that awareness of those missing factors may be crucial to successfully learning an optimal policy. For example, the transition between observed states may not obey the *markov property* unless $X_2$ is observed, the best action may depend upon whether $X_3$ is true, or the optimal policy may sometimes involve performing $a'$. The next sections aims to answer two main questions. First, by what mechanisms can an agent discover and overcome its own unawareness by asking for help? Second, when an agent discovers a new state variable or action, how can they integrate it into their current model while conserving what they have learned from past experience?

[3]This assumption, while reasonable, may always not hold (E.g., an agent may lean on a button while unaware that the button is part of the task).

## 3.1 EXPERT GUIDANCE

Our agent can expand its awareness via advice from an expert. Teacher-apprentice learning is common in the real world, as it allows learners to receive contextually relevant advice which may inform them of new concepts they would not otherwise encounter.

This paper assumes the expert has full knowledge of the true MDP, is cooperative, and infallible. Further, we abstract away the complexity of grounding natural language statements in a formal semantics and instead assume that the agent and expert communicate via a prespecified formal language (though see e.g., Zettlemoyer and Collins [2007] for work on this problem). We do not, however, assume the expert knows the agent's *current beliefs* about the decision problem.

As in Innes and Lascarides [2019], we use a minimal set of communicative acts that allow interaction between the agent and expert to proceed analogously to human teacher-apprentice interactions. Concretely, this means we want our system to have two properties. First, the expert should, for the most part, allow the agent the opportunity to learn by themselves, interjecting only when the agent is performing sufficiently poorly, or when the agent explicitly asks for advice. Secondly, following the gricean maxims of conversation [Grice, 1975], the expert should provide *non-exhaustive* answers to queries, giving just enough information to resolve the agent's current query. We want this because in real world tasks with human experts, it may be impossible to explain all details of a problem due to the cognitive constraints of the expert or costs associated with communication.

The next sections identify three types of advice whose combination guarantee the agent behaves optimally in the long run, regardless of initial awareness.[4]

### 3.1.1 BETTER ACTION ADVICE

If the expert sees the agent perform a sub-optimal action, it can tell the agent a better action it could have taken instead. For example: "When it is raining, take your umbrella instead of your sun hat". Our goal is to avoid incessantly interrupting the agent each time it makes a mistake, so we specify the following conditions for when the agent is performing sufficiently poorly to warrant correc-

[4]Full details of the syntax/semantics used by the agent and expert to communicate is in the technical supplement.

tion: Let $t$ be the current (global) time step corresponding to the $m$th step in the $nth$ episode. Similarly, let $t'$, $m'$, $n'$ be the time the expert last uttered advice. When (10-12) hold, the expert utters advice of the form (13):

$$t - t' > \mu \tag{10}$$

$$Err(n', n) > \beta \vee m > \kappa \tag{11}$$

$$\exists a' \in A^+, Q_{\pi_+}(s_{m,n}, a') > Q_{\pi_+}(s_{m,n}, a_{m,n}) \tag{12}$$

$$Q_{\pi_+}(w^s_{m,n}, a') > Q_{\pi_+}(w^s_{m,n}, a_{m,n}) \tag{13}$$

Equation (10) ensures some minimum time $\mu$ has passed since the expert last gave advice. Equation (11) ensures the expert won't interrupt unless its estimate of the agent's policy error is above some threshold $\beta$, or if the agent is unable to reach a terminal state after some reasonable bound $\kappa$ (which is required because the agent's unawareness of $A^+$ may mean its current $\epsilon$-greedy policy is not proper). If episode $n$ is unfinished, the expert estimates the expected return via the heuristic $G^n \approx \sum_{i=0}^{m-1} \gamma^i r_{i,n} + \gamma^m V_{\pi_+}(s_{m,n})$, i.e., we optimistically assume the agent will follow $\pi_+$ from now on. Taken together, $\mu, \kappa$ and $\beta$ describe the expert's *tolerance* towards the agent's mistakes. Finally, (12) ensures a better action $a'$ actually exists at this time step.

Equation (13) is the expert's utterance, and the term $w^s_{m,n}$ in it requires explanation. On first thought, the expert should utter $Q_{\pi_+}(s_{m,n}, a') > Q_{\pi_+}(s_{m,n}, a_{m,n})$, explicitly stating the full description of $s_{m,n}$. However, remember that the agent's awareness, $\mathcal{X}^t$, may be a tiny subset of $\mathcal{X}^+$. Uttering such advice may involve enumerating a huge number of variables the agent is currently unaware of. This is exactly the type of exhaustive explanation we wish to avoid, since such an explanation may place a cognitive burden on the expert, or confuse a learner. Conversely, we could instead have our expert project its intended utterance onto only those variables $\mathcal{X}^e$ for which the expert has explicit evidence the agent is aware of them (i.e., utter: $Q_{\pi_+}(s_{m,n}[\mathcal{X}^e], a') > Q_{\pi_+}(s_{m,n}[\mathcal{X}^e], a_{m,n})$). This can be understood by the agent without being made aware of any new variables, but might violate our assumption that the expert is truthful. For example, if $\exists s', s'[\mathcal{X}^e] = s_{m,n}[\mathcal{X}^e]$, but $Q_{\pi_+}(s', a) > Q_{\pi_+}(s', a')$.

The solution is to use a *sense ambiguous term* $w^s$, whose *intended* denotation is the true state $s$ (i.e $[\![w_s]\!] \in v(\mathcal{X}^+)$), but whose *default* interpretation by the agent is $s[\mathcal{X}^t]$. In words, it is as if the expert says *"In the last step, it would have been better to do $a'$ than $a_{m,n}$"*.

Thus, by introducing ambiguity, the agent can interpret the advice in two ways. The first is as a *partial description* of the true problem, which is *monotonically* true regardless of what it learns in future. On hearing (13), the agent adds (14-15) to its knowledge:

$$a' \in A^+ \tag{14}$$

$$\exists s, s[\mathcal{X}^t] = s_{m,n}[\mathcal{X}^t] \wedge Q_{\pi_+}(s, a') > Q_{\pi_+}(s, a_{m,n}) \tag{15}$$

Additionally however, the agent can choose to add its current *default interpretation* of the advice to its accumulated knowledge:

$$Q_{\pi_+}(s[\mathcal{X}^t], a') > Q_{\pi_+}(s[\mathcal{X}^t], a) \tag{16}$$

The agent can then act on the expert's advice directly by choosing $a'$ whenever $s[\mathcal{X}^t] = s_{m,n}[\mathcal{X}^t]$, regardless of what seems likely from $D_{0:t}$. We can see that even with a cooperative and infallible expert, even abstracting away issues of grounding natural language, misunderstandings can still happen due to differences in agent and expert awareness. As the next section shows, such misunderstandings can reveal gaps in the agent's awareness and help to articulate queries whose answers guarantee the agent expands its awareness.

Lemma 2 guarantees the expert's advice strategy reveals unforeseen actions to the agent so long as its performance in trials exceeds the expert's tolerance.[5]

**Lemma 2.** *Consider an* FMDP *where $\pi_+$ is proper, an agent with awareness $\mathcal{X}^t \subseteq \mathcal{X}^+, A^t \subset A^+$, and expert acting with respect to (10-13). If $\exists a \in image(\pi_+), a \notin A^t$ then as $k \to \infty$, either $Err(t, t+k) \to c$ with $c \leq \beta$ or the expert utters (12) such that $a' \notin A^t$.*

### 3.1.2 RESOLVING MISUNDERSTANDINGS

We noted before that the agent's defeasible interpretation of expert advice could result in misunderstandings. To illustrate, suppose the agent receives advice (17) and (18) at times $t - k$ and $t$:

$$Q_{\pi_+}(w^s_{t-k}, a) > Q_{\pi_+}(w^s_{t-k}, a') \tag{17}$$

$$Q_{\pi_+}(w^s_t, a) < Q_{\pi_+}(w^s_t, a') \tag{18}$$

While the intended meaning of each statement is true, the agent's default interpretations of $w^s_{t-k}$ and $w^s_t$ may be identical. That is, $s_{t-k}[\mathcal{X}^t] = s_t[\mathcal{X}^t]$. From the agent's perspective, (17) and (18) conflict, and thus give the agent a clue that its current awareness of $\mathcal{X}^+$ is deficient. To resolve this conflict, the agent asks (19) (in

[5]Proofs of lemmas/theorems are in the technical supplement

words, *"which $X$ has distinct values in $s_{t-k}$ and $s_t$?"*) and receives an answer of the form (20):

$$?\lambda X(X \in \mathcal{X}^+ \wedge s_{t-k}[X] \neq s_t[X]) \quad (19)$$

$$X \in \mathcal{X}^+ \quad (20)$$

Notice there may be multiple variables in $\mathcal{X}^+ \setminus \mathcal{X}^t$ whose assignments differ in $s_{t-k}$ and $s_t$. Thus, the expert's answer can be *non-exhaustive*, providing the minimum amount of information to resolve the agent's conflict without necessarily explaining all components of the task. This means the agent must abandon its previous defeasible interpretation of (16), but can keep (14-15), as these are true regardless of known variables. Lemma 3 guarantees the expert will reveal new state variables, provided such misunderstandings can still arise.

**Lemma 3.** *Consider an* FMDP *where $\pi_+$ is proper and an agent with awareness $\mathcal{X}^t \subset \mathcal{X}^+, image(\pi_+) \subseteq A^t \subseteq A^+$. If $\exists s \exists s' \neq s, s[\mathcal{X}^t] = s'[\mathcal{X}^t]$, and $\pi_+(s) \neq \pi_+(s')$, then as $k \to \infty$, either $Err(t, t+k) \to c$ $(c \leq \beta)$, or the expert utters (20) such that $X \notin \mathcal{X}^t$*

### 3.1.3 "IMPOSSIBLE" REWARDS

In typical FMDPs (where the agent is assumed fully aware of $\mathcal{X}^+, A^+$, and $scope_+(\mathcal{R})$), we tend only to think of the trials as providing *counts*, but for an unaware agent, a trial $d_t = \langle s_t, a_t, s_{t+1}, r_{t+1} \rangle$ also encodes *monotonic information*:

$$\exists s, s[\mathcal{X}^t] = s_{t+1} \wedge \mathcal{R}_+(s) = r_{t+1} \quad (21)$$

This constrains the form of $\mathcal{R}$ the agent must learn. Recall that $scope_t(\mathcal{R})$, may be only a subset $scope_+(\mathcal{R})$, so it might be impossible to construct an $\mathcal{R} : v(scope_t(\mathcal{R})) \to \mathbb{R}$ satisfying all descriptions (21) gathered so far. Further, those extra variables in $scope_+(\mathcal{R}) \setminus scope_t(\mathcal{R})$ may not be in $\mathcal{X}^t$. To resolve this, if the agent fails to construct a valid reward function, it asks (22) (in words, *"which variable X (that I don't already know) is in $scope(\mathcal{R})$?"*), receiving an answer (23):

$$?\lambda X(X \in scope_+(\mathcal{R}) \bigwedge_{X' \in scope_t(\mathcal{R})} X \neq X') \quad (22)$$

$$X \in scope_+(\mathcal{R}) \wedge X \in \mathcal{X}^+ \quad (23)$$

Again, the agent may be unaware of many variables in $scope_+(\mathcal{R})$, so (23) may be *non exhaustive*. Even so, we can guarantee that the agent's learned reward function eventually equals $\mathcal{R}_+$:

**Lemma 4.** *Consider an* FMDP *where $\pi_+$ is proper and an agent with awareness $A^t \subseteq A^+$, $\mathcal{X}^t \subseteq \mathcal{X}^+$, $scope_t(\mathcal{R}) \subseteq scope_+(\mathcal{R})$. As $k \to \infty$, there exists a $K$ such that for all $k \geq K$, $\mathcal{R}_{t+k}(s) = \mathcal{R}_+(s)$ for all states $s$ reachable using $A^t$.*

## 3.2 ADAPTING THE TRANSITION FUNCTION

Section 3.1 showed three ways the agent could expand its awareness of $\mathcal{X}$, $A$, and $scope(\mathcal{R})$. If we wish to improve on the naive approach of restarting learning when faced with such expansions, we must now specify how the agent adapts $\mathcal{T}$ and $\mathcal{R}$ to such discoveries.

Adapting $\mathcal{T}$ upon discovering a new action $a'$ at time $t$ is simple: Since the agent hasn't performed $a'$ in any previous trial, it can just create a new DBN, $dbn_{a'}$, using the priors outlined in section 2.2. Our new model at time $t$ then becomes $\mathcal{T} = \{dbn^t_{a_1}, \ldots dbn^t_{a_n}\} \cup \{dbn_{a'}\}$.

The more difficult issue is adapting $\mathcal{T}$ upon discovering a new state variable $Z$. The main problem is that the agent's current distributions over DBNs no longer cover all possible parent sets for each variable, nor all DTs. For example, the current distribution over $Pa^a_{X'}$ does not include the possibility that $Z$ is a parent of $X'$. Worse, since we assume in general that the agent *cannot observe $Z$'s past values* in $D_{0:t}$, it cannot observe the true value of $N^a_{Z=i|j}$, nor $N^a_{X=i|Pa^a_{X'}=j}$ when $Z \in Pa^a_{X'}$. The $\alpha$-parameters involving $Z$ are also undefined, yet we need them to calculate structure probabilities (5, 7) and parameters via (9).

The problem is that new variables make the size of each (observed) state *dynamic*, in contrast to standard problems where they are *static* (e.g., $\langle X_1 = 0, X_2 = 1 \rangle$ becomes $\langle X_1 = 0, X_2 = 1, Z = ? \rangle$ ) . We could phrase this as a missing data problem: $Z$ was hidden in the past but visible in future states, so treat the problem as a POMDP and estimate missing values via e.g., *expectation maximization* [Friedman, 1998]. However, such methods commit us to costly passes over the full state-action history, and make it hard to learn DT structures with enough sparseness to ensure a compact value function. Alternatively, we could ignore states with missing information when counts involving $Z$ are required. For example, we could use $P(Pa^a_{X'}|D_{t:n})$ to score $Pa^a_{X'}$ when $Z \in Pa^a_{X'}$ but use $P(Pa^a_{X'}|D_{0:n})$ when $Z \notin Pa^a_{X'}$. However, as Friedman and Goldszmidt [1997] points out, most structure scores, including (5), assume we evaluate models with respect to the *same data*. If two models are compared using different data sets (even if they come from the same underlying distribution), the learner tends to favour the model evaluated with the smaller amount of data. Instead, our method discards the data gathered dur-

ing the learner's previous deficient view of the hypothesis space, but *conserves* the relative *posterior* probabilities learned from past data to construct new *priors* for the $Pa^a$, $\text{DT}^a$ and $\theta^a$ in the expanded state space.

#### 3.2.1 PARENT SET PRIORS

On discovering $Z$, the agent must update $P(Pa^a_{X'})$ for each $X \neq Z$ and $a \in A^t$ to include parent sets containing $Z$. In (24) we construct a new prior $P'(Pa^a_{X'})$ using the old posterior:

$$P'(Pa^a_{X'}) = \begin{cases} (1-\rho)P(Pa^a_{X'}|D_{0:t}) & \text{if } Z \notin Pa^a_{X'} \\ \rho P((Pa^a_{X'} \setminus Z)|D_{0:t}) & \text{otherwise} \end{cases} \tag{24}$$

This preserves the likelihoods among the parent sets that do not include $Z$. Also, since $\rho < 0.5$, (24) still maintains our bias towards structures with fewer parents by initially making parent sets which do include $Z$ less probable than those that don't. To define $P(Pa^a_{Z'})$—the distribution over parent sets for the newly discovered variable $Z$—we default to (6), since the agent has no evidence (yet) concerning $Z$'s parents.

#### 3.2.2 TREE AND PARAMETER PRIORS

We must also update $P(\text{DT}^a_X|Pa^a_{X'})$ and $P(\theta^a_{X,Pa^a_{X'}}|Pa^a_{X'})$ to accommodate $Z$. Here, we return to the issue of the counts $N^a_{i|j}$ and the associated $\alpha$-parameters. As mentioned earlier, we wish to avoid the complexity of estimating $Z$'s past values. Instead, we *throw away* the past counts of $N^a_{i|j}$, but *retain* the relative likelihoods they gave rise to by packing these into new $\alpha$-parameters, as shown in (25-26):

$$\alpha^a_{X=i|Y=j} := \begin{cases} \frac{K}{|v(Z \cup Y)|} & \text{if } X = Z \\ \frac{K}{|v(Y)|} P(i, j[Y \setminus Z]|dbn^t_a) & \text{else} \end{cases} \tag{25}$$

$$P'(\text{DT}^a_{X'}|Pa^a_{X'}) \propto \prod_{\ell \in leaves(\text{DT}^a_{X'})} \beta(\alpha^a_{X'=1|\ell}, \ldots, \alpha^a_{X'=n|\ell}) \tag{26}$$

Equation (25) summarizes $D_{0:t}$ via inferences on the old best DBNs, then encodes these inferences in the new $\alpha$-parameters. The revised $\alpha$-parameters ensure the new tree structure prior and expected parameters defined via (26) and (9) bias towards models the agent previously thought were likely. Indeed, the larger the (user-specified) $K$ parameter is, the more the distributions learned *before* discovering $Z$ influence the agent's reasoning *after* discovering $Z$.

### 3.3 ADAPTING REWARD AND VALUE TREES

On becoming aware that $Z$ is part of $scope_+(\mathcal{R})$, the agent may wish to restructure its reward tree. This is because awareness that $Z \in scope_+(\mathcal{R})$ means there are tests of the form $Z = i$ that the agent has not yet tried which may produce a more compact tree. In the language of ITI, the current test nodes are "stale", and must be rechecked to see if a replacement test would yield a tree with better information gain. If the agent was unaware of $Z$ (i.e, $Z \notin \mathcal{X}^t$), we can still test on assignments to $Z$ by following the ITI convention that any state where $Z$ is missing automatically fails any test on $Z$.

Once we have updated $\mathcal{T}$ and $\mathcal{R}$, there is no need to make further changes to $V_t$ in response to a new action $a'$ or variable $Z$. In effect, this encodes our conservative intuition that the true $V_{\pi_+}$ is more likely to be closer to the agent's current estimate $V_t$ than some arbitrary value function. The agent essentially assumes (in absence of further information) that the value of a state is indifferent to this newly discovered factor. In subsequent trials where the agent performs $a'$ or observes $Z$, Algorithm 1 ensures information about this new factor is incorporated into the agent's value function.

**Algorithm 2** Learning FMDPs with Unawareness

**function** LEARNFMDPU($A^0$, $\mathcal{X}^0$, $\mathcal{T}_0$, $Q_0$, $V_0$, $s_0$)
  **for** $t = 1 \ldots maxTrials$ **do**
    $\langle s_t, r_t \rangle \leftarrow \epsilon$-GREEDY($s_{t-1}$, $Q_{t-1}$, $adv_{0:t-1}$)
    $\langle \mathcal{T}_t, \mathcal{R}_t \rangle \leftarrow$ Add $\langle s_t, r_t \rangle$ via (5-9) & ITI
    **if** Update to $\mathcal{R}_t$ fails **then**
      $Z \leftarrow$ Ask expert (19)
      $\langle scope_t(\mathcal{R}), \mathcal{X}^t \rangle \leftarrow$ Append $Z$ to each
      $\mathcal{R}_t \leftarrow$ Update via ITI
    **if** (10-12) are true **then**
      $adv_t \leftarrow$ Expert advice of form (13)
      **if** $adv_t$ mentions action $a' \notin A^{t-1}$ **then**
        $\mathcal{A}^t \leftarrow \mathcal{A}^{t-1} \cup \{a'\}$
        $\mathcal{T}_t \leftarrow \mathcal{T}_{t-1} \cup \{dbn_{a'}\}$ made via (6)
      **if** $adv_{0:t-1}$ conflicts with $adv_t$ **then**
        $Z \leftarrow$ Ask expert (19)
        $\mathcal{X}^t \leftarrow \mathcal{X}^{t-1} \cup \{Z\}$
    **if** $\mathcal{X}^{t-1} \neq \mathcal{X}^t$ **then**
      $\mathcal{T}_t \leftarrow$ Update via (25, 5, 26, 7, 9)
    $\langle V_t, Q_t \rangle \leftarrow$ INCSVI($\mathcal{R}_t, \mathcal{T}_t, \mathcal{V}_{t-1}$)

Algorithm 2 outlines how the agent updates $\mathcal{T}$, $\mathcal{R}$, and $V$ in response to new data and expert advice. Given Algorithm 2, Theorem 1 guarantees our agent behaves indistinguishably from a near-optimal policy in the long run, regardless of initial awareness (provided all $X \in \mathcal{X}^+$ are relevant to expressing the optimal policy).

**Theorem 1.** *Consider an* FMDP *where $\pi_+$ is proper and an agent with initial awareness $\mathcal{X}^0 \subseteq \mathcal{X}^+, A^0 \subseteq A^+$, and $scope_0(\mathcal{R}) \subseteq scope_+(\mathcal{R})$ acts according to Algorithm 2. If for all $X \in \mathcal{X}^+$, there exists a pair of states $s, s'$ such that $s[\mathcal{X}^+ \setminus X] = s'[\mathcal{X}^+ \setminus X]$, $s[X] \neq s'[X]$, and $\pi_+(s) \neq \pi_+(s')$, then as $t \to \infty$, $Err(0,t) \to c$ such that $c \leq \beta$.*

## 4 EXPERIMENTS AND RESULTS

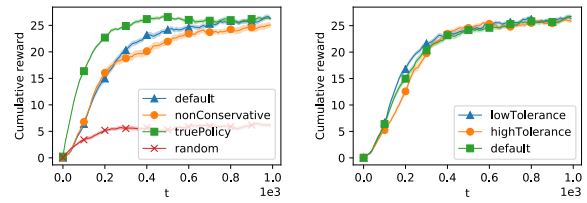
Our experiments show that agents following Algorithm 2 converge to near-optimal behaviour in both theory and practice. Further, we show that conserving information on $\mathcal{T}$ and $V$ gathered before each new discovery allows our agent learn faster than one which abandons this information. We do not investigate assigning an explicit budget to agent-expert communication, leaving this to future work. However we do show how varying the expert's tolerance affects the agent's performance.

We test agents on two well-known problems: *Coffee-Robot* and *Factory*.[6] In each, our agent begins with only partial awareness of $\mathcal{X}^+$, $A^+$ and $scope_+(\mathcal{R})$. The agent takes actions for $T$ time steps, using an $\epsilon$-greedy policy ($\epsilon = 0.1$). When the agent enters a terminal state, we reset it to one of the initial states randomly. We use the *cumulative reward* across all trials as our evaluation metric, which acts as a proxy for the quality of the agent's policy over time. To make the results more readable, we apply a discount of 0.99 at each step, resulting in the metric $R_t^{disc} = r_t + 0.99 * R_{t-1}^{disc}$.

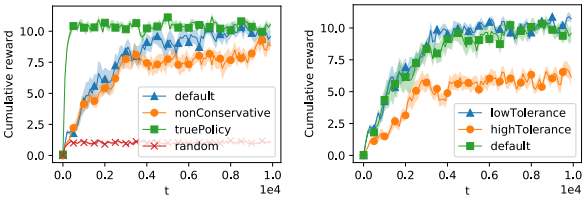
We test several variants of our agent to show the effectiveness of our approach. The **default** agent follows Algorithm 2 as is, with parameters $\rho = 0.1$, $K = 5.0$, $\mu = 10$, $\beta = 0.1$, $\kappa = 50$ in equations (6), (24), (25), and (10-12) respectively. The **nonConservative** agent does not conserve information about $V$, nor $\mathcal{T}$ via (24-26) when a new factor is discovered. Instead, it resets $V$ and $\mathcal{T}$ to their initial values. This agent is included to show the value of conserving past information as $\mathcal{X}$ and $A$ expand. The **truePolicy** and **random** agents start with full knowledge of the true FMDP, and execute an $\epsilon$-greedy version of $\pi_+$, or a choose random action respectively. These agents provide an upper/lower bound on performance. The **lowTolerance / highTolerance** agents change the expert's tolerance to $\beta = 0.01$ and $\beta = 0.5$.

### 4.1 COFFEE ROBOT

*Coffee-Robot* is a small sequential problem where a robot must purchase coffee from a cafe, then return it to their owner. Also, the robot gets wet if it has no umbrella when it rains. The problem has 6 boolean variables—HUC (user has coffee), HRC (robot has coffee), R (raining), W (wet), L (location), U (umbrella)—and 4 actions—MOVE, DELC, BUYC and GETU— making 256 state/action pairs. The terminal states are those where HUC $= 1$; initial states are all non-terminal ones. Our agent has initial awareness $A^0 = \{\text{MOVE}\}$, $\mathcal{X}^0 = scope_0(\mathcal{R}) = \{\text{HUC}\}$ and discount factor $\gamma = 0.8$.[7]




(a) *Coffee Robot* ($T = 1000$. Average of 50 experiments)




(b) *Factory* ($T = 10000$. Average of 20 experiments)

Figure 2: Cumulative Rewards. Shaded areas represent standard error from the mean.

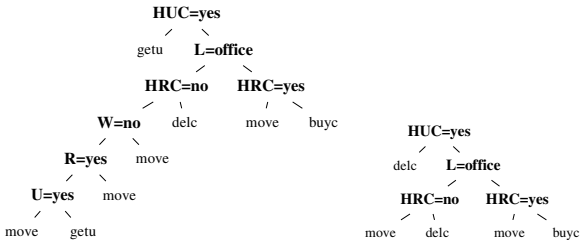
Figure 2a shows each agent's (discounted) cumulative reward. Despite starting unaware of factors critical to success, the default agent quickly discovers the relevant actions and states with the expert's aid, and converges on the optimal policy. The non-conservative agent also learns the optimal policy, but takes longer. This shows the value of conserving $\mathcal{T}$ and $V$ on discovering new state variables. We also see how expert tolerance affects performance. The agent paired with high tolerance expert learns a (marginally) worse final policy, but this makes little difference to cumulative reward. Figure 3 shows why: The agent learned a "good enough" policy, so the expert doesn't reveal the "get umbrella" (GETU) action, which yields only a minor increase in reward. Figure 4 supports this explanation, showing how more tolerant experts reveals less variables over time.

### 4.2 FACTORY

*Factory* is a larger problem ($|A^+| = 14$, $\mathcal{X}^+ = 14$, 774144 state/action pairs), which shows our method works on more realistically sized tasks. Here, an agent

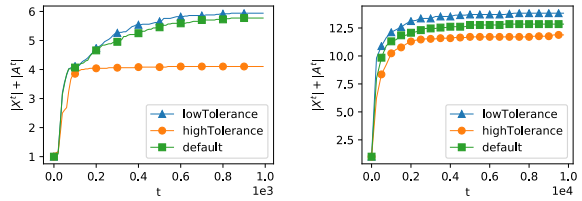
[6] Full specifications at `https://cs.uwaterloo.ca/~jhoey/research/spudd/index.php`

[7] Original setting was $\gamma = 0.9$. Changed to make $\pi_+$ proper.




(a) Default Tolerance (b) High Tolerance

Figure 3: Typical final policy depending on tolerance




(a) *Coffee Robot* task (b) *Factory* task

Figure 4: Awareness of $|\mathcal{X}^+|$ and $|A^+|$

must shape, paint and connect two widgets to create products of varying quality. Some actions (like bolting) produce high quality products, whereas others (like gluing) produce low quality products. The agent receives a higher reward for producing goods which match the demanded quality.[8]. The terminal states are those where CONNECTED $= 1$; initial states are non-terminals where it is possible to connect two components. Our agent's initial awareness is $\mathcal{X}^0 = scope_0(\mathcal{R}) = \{\text{CONNECTED}\}$, $A^0 = \{\text{BOLT, GLUE, DRILLA, DRILLB}\}$, with $\gamma = 0.9$. This represents a simplified task where the agent thinks the only goal is connecting the widgets.

Figure 2b shows results similar to previous experiments. The default agent converges on optimal behaviour, and does so quicker than the non-conservative agent. Varying the expert's tolerance now has a larger effect on the rate at which factors are discovered and on convergence towards the optimal policy, because there are now many more unforeseen variables/actions the agent can discover. A full breakdown of which expert messages were sent when is available in the technical supplement.

## 5 RELATED WORK

A related notion to unawareness is *imprecise probability* [Delgado et al., 2011]. Such work allows you to do inference even when you are unable to learn a unique probability distribution from the available evidence. However, such methods still rely on knowing which distributions are *possible* and thus cannot handle the notion of unforeseen concepts as outlined in this paper.

Rong [2016] define *markov decision processes with unawareness* (MDPUs) to learn optimal behaviour when starting unaware of some actions. They apply MDPUs to a robotic-motion problem with around 1000 discretised atomic states. The agent uses an abstract *explore* move, which randomly reveals useful motions they were previously unaware of. Our work differs from theirs in several ways. First, we provide a concrete mechanism for discovering unforeseen factors via expert advice. Second, our agent discovers *explicit state variables* rather than atomic states, and focusses on exploiting the inherent structure in problems with many features. This enables us to scale up to complex decision problems, where the agent converges on an optimal policies in a (true) state space around a million atomic states, as opposed to around 1000. McCallum and Ballard [1996] also learn an increasingly complex representation of the state space by gradually distinguishing between states which yield different rewards. Rather than dealing with unawareness, their approach focusses on *refining* an existing state space. They do not support introducing *unforeseen* states or actions the learner was unaware of before learning.

Many works use expert interventions to improve performance via reward shaping [Knox and Stone, 2009], corrections [Torrey and Taylor, 2013], preferences [Kunapuli et al., 2013], and probabilistic logic [Odom and Natarajan, 2016]. Yet all such methods assume the expert's intended meaning can be understood without expanding the agent's current state and action space. Our work allows experts to utter advice where ambiguity arises from their greater awareness of the problem.

## 6 CONCLUSION

We have presented an agent-expert framework for learning optimal behaviour in both small and large FMDPs, even when one starts unaware of factors critical to success. Further, we showed that conserving one's beliefs improves the quality of learning. In future work, we aim to lift some assumptions imposed on the expert, and expand the expressiveness of its advice. For instance, we could let the expert be fallible, or allow questions on the *structure* of $\mathcal{T}$ [Masegosa and Moral, 2013].

## ACKNOWLEDGEMENTS

This work is supported by EPSRC (UK) and the Alan Turing Institute.

[8] Rewards were scaled to range 0.0-1.0 and, to make $\pi_+$ proper, terminal states were given a reward of at least 0.01.

## References


M. Araya-López, O. Buffet, V. Thomas, and F. Charpillet. Active learning of MDP models. In *European Workshop on Reinforcement Learning*, pages 42–53. Springer, 2011.

C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 115–123. Morgan Kaufmann Publishers Inc., 1996.

C. Boutilier, R. Dearden, and M. Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial Intelligence*, 121(1):49–107, Aug. 2000. ISSN 0004-3702. doi: 10.1016/S0004-3702(00)00033-3.

M. Cakmak and A. Thomaz. Designing robot learners that ask good questions. *Proceedings of the 7th Annual ACM/IEEE International Conference on Human-Robot Interaction*, 2012.

A. Coenen, J. D. Nelson, and T. M. Gureckis. Asking the right questions about human inquiry. *Open-Coenen, Anna, Jonathan D Nelson, and Todd M Gureckis."Asking the Right Questions About Human Inquiry". PsyArXiv*, 13, 2017.

T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational intelligence*, 5(2):142–150, 1989.

T. Degris and O. Sigaud. Factored markov decision processes. *Markov Decision Processes in Artificial Intelligence*, pages 99–126, 2010.

T. Degris, O. Sigaud, and P.-H. Wuillemin. Learning the structure of factored markov decision processes in reinforcement learning problems. In *Proceedings of the 23rd international conference on Machine learning*, pages 257–264. ACM, 2006.

K. V. Delgado, S. Sanner, and L. N. De Barros. Efficient solutions to factored MDPs with imprecise transition probabilities. *Artificial Intelligence*, 175(9-10):1498–1527, 2011.

F. Doshi-Velez. The infinite partially observable Markov decision process. In *Advances in neural information processing systems*, pages 477–485, 2009.

N. Friedman. The Bayesian structural EM algorithm. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 129–138. Morgan Kaufmann Publishers Inc., 1998.

N. Friedman and M. Goldszmidt. Sequential update of Bayesian network structure. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 165–174. Morgan Kaufmann Publishers Inc., 1997.

H. P. Grice. Logic and conversation. *1975*, pages 41–58, 1975.

C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

C. Innes and A. Lascarides. Learning Structured Decision Problems with Unawareness. In *International Conference on Machine Learning*, pages 2941–2950, 2019.

W. B. Knox and P. Stone. Interactively shaping agents via human reinforcement: The TAMER framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16. ACM, 2009.

G. Kunapuli, P. Odom, J. W. Shavlik, and S. Natarajan. Guiding autonomous agents to better behaviors through human advice. In *2013 IEEE 13th International Conference on Data Mining*, pages 409–418. IEEE, 2013.

A. R. Masegosa and S. Moral. An interactive approach for Bayesian network learning using domain/expert knowledge. *International Journal of Approximate Reasoning*, 54(8):1168–1181, Oct. 2013. ISSN 0888-613X. doi: 10.1016/j.ijar.2013.03.009.

A. K. McCallum and D. Ballard. *Reinforcement learning with selective perception and hidden state*. PhD Thesis, University of Rochester. Dept. of Computer Science, 1996.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, and others. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

P. Odom and S. Natarajan. Actively interacting with experts: A probabilistic logic approach. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 527–542. Springer, 2016.

N. Rong. *Learning in the Presence of Unawareness*. PhD Thesis, Cornell University, 2016.

R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

L. Torrey and M. Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1053–1060. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

P. E. Utgoff, N. C. Berkman, and J. A. Clouse. Decision tree induction based on efficient tree restructuring. *Machine Learning*, 29(1):5–44, 1997.

L. Zettlemoyer and M. Collins. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.